Influence of Social Circles on User Recommendations

Chaitanya Krishna Kasaraneni

San Jose State University

INFLUENCE OF SOCIAL CIRCLES ON USER RECOMMENDATIONS

A Thesis

Presented to

The Faculty of the Department of Computer Engineering

San José State University

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

by

Chaitanya Krishna Kasaraneni

December 2020

The Designated Thesis Committee Approves the Thesis Titled

INFLUENCE OF SOCIAL CIRCLES ON USER RECOMMENDATIONS

by

Chaitanya Krishna Kasaraneni

APPROVED FOR THE DEPARTMENT OF COMPUTER ENGINEERING

SAN JOSÉ STATE UNIVERSITY

December 2020

Mahima Agumbe Suresh, Ph.D.	Department of Computer Engineering
Magdalini Eirinaki, Ph.D.	Department of Computer Engineering
Gheorghi Guzun, Ph.D.	Department of Computer Engineering

ABSTRACT

INFLUENCE OF SOCIAL CIRCLES ON USER RECOMMENDATIONS

by Chaitanya Krishna Kasaraneni

Recommender systems are powerful tools that filter and recommend content relevant to a user. One of the most popular techniques used in recommender systems is collaborative filtering. Collaborative filtering has been successfully incorporated in many applications. However, these recommendation systems require a minimum number of users, items, and ratings in order to provide effective recommendations. This results in the infamous cold start problem where the system is not able to produce effective recommendations for new users. In recent times, with escalation in the popularity and usage of social networks, people tend to share their experiences in the form of reviews and ratings on social media. The components of social media like influence of friends, users' interests, and friends' interests create many opportunities to develop solutions for sparsity and cold start problems in recommender systems. This research observes these patterns and analyzes the role of social trust in baseline social recommender algorithms SocialMF - a matrix factorization-based model, SocialFD - a model that uses distance metric learning, and GraphRec - an attention-based deep learning model. Through extensive experimentation, this research compares the performance and results of these algorithms on datasets that these algorithms were tested on and one new dataset using the evaluations metrics such as root mean squared error (RMSE) and mean absolute error (MAE). By modifying the social trust component of these datasets, this project focuses on investigating the impact of trust on performance of these models. Experimental results of this research suggest that there is no conclusive evidence on how trust propagation plays a major part in these models. Moreover, these models show slightly improved performance when supplied with modified trust data.

ACKNOWLEDGMENTS

I would like to thank my advisor, Dr. Mahima Agumbe Suresh for guiding me in completing this project and supporting me in overcoming all the obstacles. It was a pleasure working alongside Dr. Mahima on this research.

I would also like to thank my thesis committee members, Dr. Magdalini Eirinaki and Dr. Gheorghi Guzun for sharing their knowledge and providing helpful insights for this research.

Finally, I would like to thank my family and friends. Without their support, I would have never been able to complete this thesis.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

API - Application Programming Interface

CF - Collaborative Filtering

MAE - Mean Absolute Error

MLP - Multi-Layer Perceptron

NTU - No Trust between Users

OTD - Original Trust Data

OTT - Over-the-top

RMSE - Root Mean Squared Error

RS - Recommender/Recommendation Systems

SGD - Stochastic Gradient Descent

SVD - Singular value decomposition

UTE - Users Trust Everyone Including Themselves

UTEU - Users Trust Everyone else Excluding Themselves

UTU - Users Trust only Themselves

1 INTRODUCTION

The amount of information available over the web has become immeasurable with the proliferation of Internet usage. With this availability of huge amounts of data, it has become progressively necessary to present the users with relevant online content based on their interests. Recommender systems (RS) do this task by predicting preferences or ratings that the users would give to a set of items [4]. These ratings are predicted based on users' interests. RS are used in diverse areas such as song or playlist recommendations on music apps like Pandora and Spotify, video recommendations on over-the-top (OTT) streaming platforms like Netflix and YouTube, product recommendations on e-commerce websites like Amazon and eBay, and posts or content recommendation on social media platforms such as Twitter, Facebook, Pinterest, etc. [4]

Each RS contains a set of users and set of items, in which each user u gives a rating r to an item i. The task of the RS is to predict rating r' that user u would give to a non-rated item i' or to recommend user u with some items based on the ratings that are already given by user to other items.

Collaborative filtering (CF) is one of the most widely used techniques in recommender systems. CF has two senses [5].

1) Narrower sense: CF makes automatic predictions of users' interests based on the preferences collected from several users.
2) General Sense: CF is a technique that filters information or content using mechanisms that involve collaboration between multiple data sources, users/agents, etc. [5]

Applications of CF involve processing of very large datasets. CF techniques can be applied to different kinds of data: sensor data, financial data, e-commerce data, web application data, etc.

1.1 Types of Collaborative Filtering Recommender Systems

Collaborative Filtering RS are generally classified into the following types:

1) Memory-based CF
2) Model-based CF
3) Hybrid CF
4) Deep Learning-based CF

1.1.1 Memory-based Collaborative Filtering

Memory-based CF systems explore the user-item rating matrix and recommend based on the ratings of item *i* by a set of users whose rating profiles are most similar to that of user *u* [6].

1.1.2 Model-based Collaborative Filtering

Model-based approaches learn and only store the parameters of a model. As a result, these algorithms have no need to explore the rating matrix. Model-based approaches are fast after the algorithms learn parameters of the model.

The performance bottleneck for model-based approaches is the training phase whereas memory-based approaches have no training phase. However, the prediction is slower as user-item matrix needs to be accessed several times.

1.1.3 Hybrid Collaborative Filtering

Many applications combine both memory-based and model-based collaborative filtering algorithms. These hybrid models improve the performance of predictions and overcome the limitations of traditional collaborative filtering algorithm. Most of the commercial recommender systems are hybrid, such as Google News Recommender System [7].

1.1.4 Deep Learning-based Collaborative Filtering

In recent years, with the increase in usage of deep learning and neural networks, many deep learning-based recommendation techniques have been developed. Some of these algorithms generalize the traditional matrix factorization algorithms by utilizing nonlinear neural networks [8] or leverage other techniques like autoencoders [9]. While deep learning has been used in different scenarios of recommendations, the effectiveness of deep learning models is questioned when used in traditional collaborative filtering [10]. Analysis on papers published in top conferences such as SIGIR, KDD, WWW, RecSys, etc. shows that very few papers and articles are reproducible [10].

1.2 Cold Start Problem

The cold start problem is one of the most well known and researched problems in recommender systems. In a typical recommender system, a user *u*'s profile is compared to some reference characteristics related to his or her behavior or items. Depending on the type of recommender system, user *u* is associated with various characteristics such as ratings, interests, page visits, purchase history, etc. Cold start problem specifically occurs when a recommender system cannot infer these characteristics to user *u*. In RS, cold start users are users who are either new to the platform or have given only a few ratings. Using similarity-based approaches, it is infeasible to find corresponding similar users since the cold start users only have a few ratings.

1.2.1 Cold Start Problem and Social Networks

In present day, with the rapid increase in the popularity and usage of social networks, there is a dramatic growth in the number of registered users and various products, which also leads to an intractable increase in the cold start problem and the sparsity of datasets. Collaborative filtering works effectively when users have expressed a minimum number of ratings to have common ratings with other users in the dataset. For relatively new users, the performance suffers due to the cold start problem.

The interpersonal relationships, especially the friends circles in social networks make it possible to solve the cold start and sparsity problem. The richness of social media gives us some valuable insights to drive user recommendations, especially for items such as music, movies, news, brands, and travel. Many social network-based models for recommender systems have been developed to refine the performance but only a handful have considered social circles in their respective approaches. This gap motivates the development of an RS that considers the personal interests of users, interpersonal similarity [11] of interests with their friends, and influence of these interpersonal interests.

1.3 Social Rating Networks

A social rating network consists of a social network with ratings expressed by each user to some items apart from creating social relations to other users. A sample social rating network is depicted in Fig. 1. The icons, under users, in Fig. 1 indicate the items and the number under each icon indicates the rating that a user has given to that particular item. Table 1 shows the matrix representation of the user-item ratings, and Table 2 shows the user-user relationship. Here '1' indicates that user *u* trusts *v*. The terms "trust network" and "social network" are used synonymously in this paper.

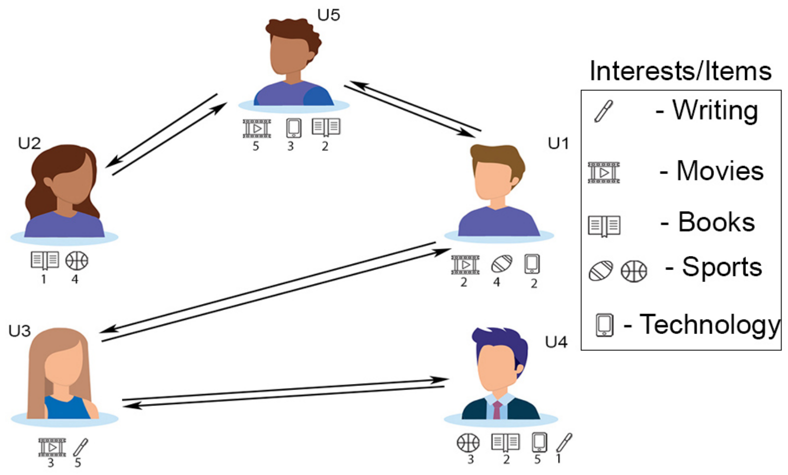


Fig. 1. Sample social rating network showing users' interests and relations between users.

Table 1
Sample User-Item Rating Matrix

	Sports	Technology	Movies	Writing	Books
U1	4	2	2		
U2	4				1
U3			3	5	
U4	3	5		1	2
U5		3	5		2

Table 2
Sample Social Trust Matrix

	U1	U2	U3	U4	U5
U1			1		1
U2					1
U3	1			1	
U4			1		
U5	1	1			

1.3.1 Social Rating Networks and Recommender Systems

A number of RS techniques have been proposed using social rating networks [12] [13] [14] [15] [16] [17]. Of these techniques, [12] [13] [16] [17] are memory-based approaches which explore a social network and find neighborhoods of direct or indirect trusted users and recommend to users by aggregating ratings. These techniques utilize transitive property for obtaining trust from indirect neighbors. These memory-based algorithms are slower compared to model-based approaches in the test phase since they have to traverse the entire social network.

Model-based RS techniques using social rating networks have been developed in [14] [15]. These techniques utilize the matrix factorization to obtain latent features for each user and item from the ratings observed. Experiments show that these model-based approaches perform better compared to state-of-the-art memory-based algorithms. But the major setback is that these algorithms do not take account of trust propagation. To solve this issue, SocialMF [1], a matrix factorization technique-based recommendation model, was proposed. This model includes trust propagation to improve the quality of recommendations. Another method called SocialFD [2] that incorporates distance metric learning alongside matrix factorization was proposed to optimize performance.

With the recent increase in use of attention, deep learning, and graph neural networks, an attention-based deep learning model known as GraphRec [3] was developed. This model contains two components. The first one is to learn user latent factors that contains two separate aggregations, one for learning interactions between users and items in the user-item graph and the other for social aggregation. The second component is extracting item latent factors which contains user aggregation. Finally, model parameters are learned via predictions by integrating both the components.

The rest of this paper is arranged as follows: Section 2 discusses some related works. Section 3 gives an overview of this research. Section 4 summarizes the SocialMF [1], SocialFD [2], and GraphRec [3] models. Section 5 explores experiment methodology and datasets used in this research. Section 6 compares experimental results. Finally, Section 7 concludes the paper with some directions for future work in Section 8.

2 RELATED WORKS

This section reviews some of the works in recommendation mechanisms that utilize a social network in addition to the user-item data. Trust propagation is widely considered in memory-based approaches whereas model-based recommendation approaches broadly use matrix factorization [18] [19] [20]. The major setback with these techniques is that they do not take the social network of users into consideration while recommending or filtering content. Model-based techniques utilize matrix factorization approach for recommending content in social networks [14] [15], but these approaches do not examine trust propagation. In this section, some model-based works in social networks are discussed after reviewing memory-based models.

Using a modified breadth-first search technique on the trust network, a memory-based algorithm called TidalTrust [12] was proposed to determine a prediction. TidalTrust tries to find users who rated particular items (raters) with the shortest distance from the user and combines their ratings weighted with trust between the user and these raters [12]. TidalTrust combines the trust value between user *u*'s direct neighbors and *v* weighted by the trust values of *u* and its direct neighbors to compute the trust value between users *u* and *v* who are indirectly connected [12].

Another approach called MoleTrust, which is similar to TidalTrust, was introduced in [16]. The major difference is that MoleTrust [16] considers all raters until maximum-depth of the input irrespective of a specific user or item. Backward exploration is used in MoleTrust to compute the trust between users *u* and *v*. For example, the calculated trust value is an aggregation of trust between user *u* and users who directly trust user *v* weighted by their direct trust values.

In [21], the authors proposed a maximum flow trust metric called Advogato. This approach helps in discovery of trusted users in an online community. Input for Advogato will be the total number of users to be trusted, *n*. The Advogato algorithm needs to understand the whole network structure in order to assign capacities to the edges of the network. Furthermore, Advogato only calculates the nodes to trust but not different degrees of trust. This technique is not suitable for trust-based recommendations as the trusted users are independent of users and items in the network and the distinction between trusted users is negligible.

To consider enough ratings and exclude noisy data, a random walk approach called TrustWalker was proposed in [13]. This approach combines both item-based and trust-based recommendations. This method not only considers ratings of the required item, but also the ratings of similar items. The likelihood of considering these similar items increases with the increase in walk length. Additionally, this framework contains both trust-based and item-based recommendations as special cases. Experiments show that this algorithm outperforms other existing memory-based techniques allowing them to calculate confidence of predictions.

In [14], authors proposed an approach called STE which is a matrix factorization-based approach for recommendations in social networks. This approach is a sequential combination of basic matrix factorization technique [20] and a social network-based technique. Experimental results show that this approach excelled the existing basic matrix factorization based recommendation techniques. However, the feature vectors of direct neighbors of user *u* affect the ratings of *u* instead of affecting the feature vector of *u* in this model [1]. Also, this model does not address trust propagation. Although a social network is integrated, real world recommendations are not reflected in this model. Furthermore, this model's interoperability is difficult as two sets of dissimilar feature vectors is considered.

In recent years, there have been many developments in deep neural networks for graph data, especially social network data [22]. These are known as Graph Neural Networks (GNNs). Works like [23] [24] [25] have been proposed to learn meaningful insights and representations for graph data. The main idea in these works is to use neural networks for aggregating features from local graph neighborhoods iteratively. Some of these models use graph neural networks. DANSER [26] is one of the most recent algorithms that uses dual graph attention networks to learn representations for two-fold social effects, where one is modeled by a user-specific attention weight and the other is modeled by a dynamic and context-aware attention weight [26].

There are some deep learning models that use AutoEncoders for dimensionality reduction in making recommendations. AutoTrustRec [27] is one such algorithm that uses AutoEncoders for social trust based recommendations. In this algorithm, a shared layer is applied on user and item layers which synchronize user-item ratings matrix and user-user trust matrix [27]. Also this algorithm utilizes both direct and indirect trust in the trust layer. The ratings and social trust are given as input to encoder and corresponding shared layer where trust and item values are encoded. These encoded values are then synchronised such that the output of encoder layer is formed. This output contains only values where user-user trust value pair matches with user-item value. Then activation function and neural networks are used to approximate any continuous functions and discontinuous functions respectively. In the decoder layer, social trust and item values are reconstructed, and predicted ratings are given as output.

There are some other social recommender models. Of them, SocialMF [1], SocialFD [2], and GraphRec [3] are focused in this paper.

3 RESEARCH OVERVIEW

The main aim of this research is to investigate how modifying social trust affects the recommendations in a social rating network. To understand this in depth, we compared the performance of some baseline algorithms, GraphRec [3], RSTE [14], SoRec [15], SoReg [28], Singular value decomposition (SVD) [19] based RS, SocialFD [2], and SocialMF [1]. Fig. 2 indicates the performance of these algorithms on Epinions, CiaoDVD, FilmTrust and TwitterEgo datasets in terms of root mean squared error (RMSE) and mean absolute error (MAE).

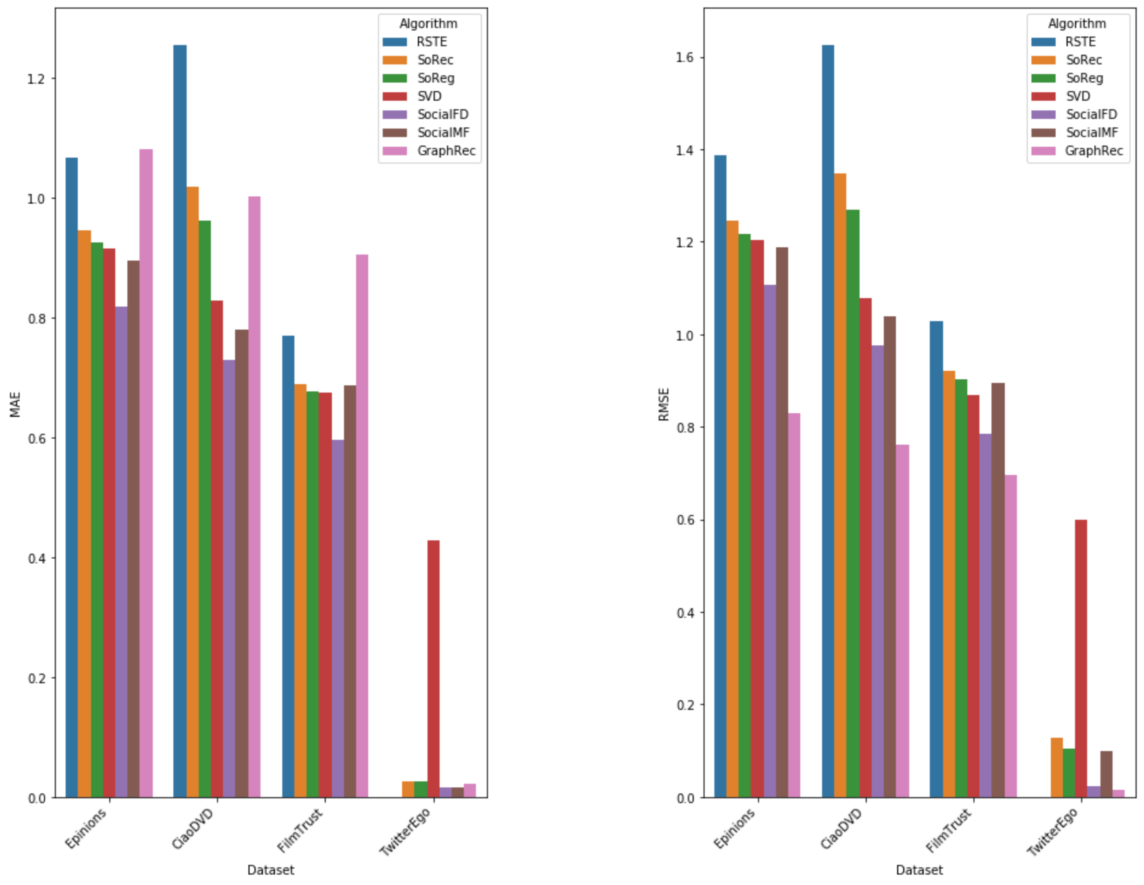


Fig. 2. Performance comparison of various recommender algorithms.

These results are obtained by performing k-Fold cross validation and taking average of errors RMSE and MAE. These algorithms performed better for lower learning rates in the range 0.001 to 0.01. The performance of social algorithms is dependent on a social regularization parameter (which is given as a hyperparameter).

Of these seven algorithms, we explore three social recommender algorithms that performed better on these datasets, SocialMF [1], SocialFD [2], and GraphRec [3], both in the presence and the absence of social trust and we also evaluate the performance of these models using metrics like MAE and RMSE. More accurately, the contributions of this project are:

- Create a new large dataset, extending a prior dataset called TwitterEgo [29], with high quality social circle information extracted from Twitter
- Perform extensive experiments on the SocialMF, SocialFD, and GraphRec algorithms with the datasets that these algorithms were tested on and the new dataset based on TwitterEgo
- Compare the results of these experiments based on the evaluation metrics including RMSE and MAE
- Investigate the performance of these models in the following cases additional to the original trust data:
  - When there is no trust between any users i.e., the number of trust statements is 0,
  - When the users are friends only with themselves i.e., users trust only themselves,
  - When the users are friends with everyone else excluding themselves, and
  - When the users are friends with everyone including themselves

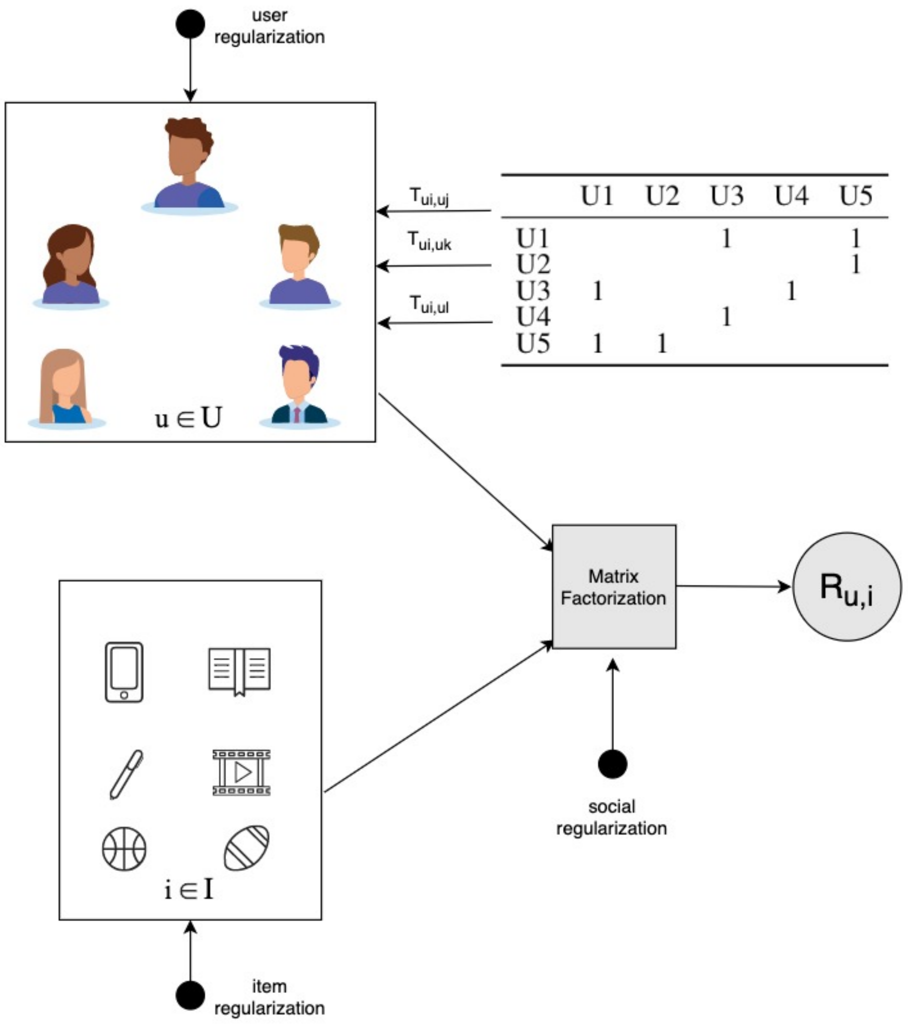
4 MODELS USED IN THIS RESEARCH

Most of the conventional recommender system algorithms do not consider the social relations among the users in a network. With the increasing usage of social networking applications, incorporating this information into recommendation systems has also become increasingly important. The following are three baseline algorithms that include these social relations and also use trust propagation in the recommendation process. We chose these algorithms because SocialMF [1] and SocialFD [2] are usually used as baselines in new trust-based recommender systems. As a new deep learning algorithm, GraphRec [3] is a representative algorithm to investigate if deep learning models are required for making predictions in recommender systems utilizing only trust information. These models are described in the rest of this section.

4.1 SocialMF Model

Jamali and Ester proposed this method in [1]. This model incorporates propagation of trust into matrix factorization for recommending a product or an item in social networks and is closely related to the STE model [14]. This model addresses trust transitivity in social networks i.e., this model considers propagation of trust. From the graphical representation of the SocialMF model in Fig. 3 [1], it is evident that feature vector of a user *u* is dependent on feature vectors of the user's direct neighbor. This is a recursive dependence, i.e., feature vector of direct neighbor depends on feature vectors of his or her direct neighbors.

In the baseline matrix factorization model [20] and the STE model [14], features are learned from observed ratings only. However, in real world social networks, most of the users only participate in social network but do not express ratings to items. This makes it hard to learn feature vectors from observed ratings. SocialMF model handles these users by learning to tune the latent features of these users close to their neighbors. Hence, even if a user does not express any ratings, the feature vectors are learned in a way that these vectors are close to feature vectors of their neighbors. As the learned features are typically based on the retained observed ratings, the evaluation of these learned features for users who haven't expressed ratings is difficult.



	U1	U2	U3	U4	U5
U1			1		1
U2					1
U3	1			1	
U4			1		
U5	1	1			

Fig. 3. Graphical representation of SocialMF model [1].

In Fig. 3, u represents a user belonging to the user set U and i represents an item belonging to the item set I respectively. The term T_{u_i,u_j} represents trust between user u_i

and user u_j. $R_{u,i}$ denotes the predicted rating that user u would give to an item i. The user latent factors are calculated using the formula:

$$\hat{U_u} = \frac{\Sigma_{v \varepsilon N_u} T_{u,v} U_v}{\mid N_u \mid} \tag{1}$$

where $\hat{U_u}$ is the calculated latent factor of user u based on the known latent feature vectors of direct neighbors of u denoted by N_u. $T_{u,v}$ denotes trust between user u and v and U_u denotes the user u's original latent vector.

The loss function used in SocialMF is similar to that of SVD with an addition of social regularization term λ_T. The loss function can be represented as:

$$L = \text{SVD loss} + \text{regularization terms} + \lambda_T/2 * \text{social graph loss}, \tag{2}$$

where λ_T is a parameter to tune the influence of the trust matrix on the recommendation.

In a social network, some users actively participate in rating a product or writing a review, but most of the users express very few ratings. These users are called cold-start users. This algorithm has shown improved performance on cold-start users compared to the STE [14] model. However, the SocialMF model has higher cost in calculation of social factor and its gradients against user and item feature vectors.

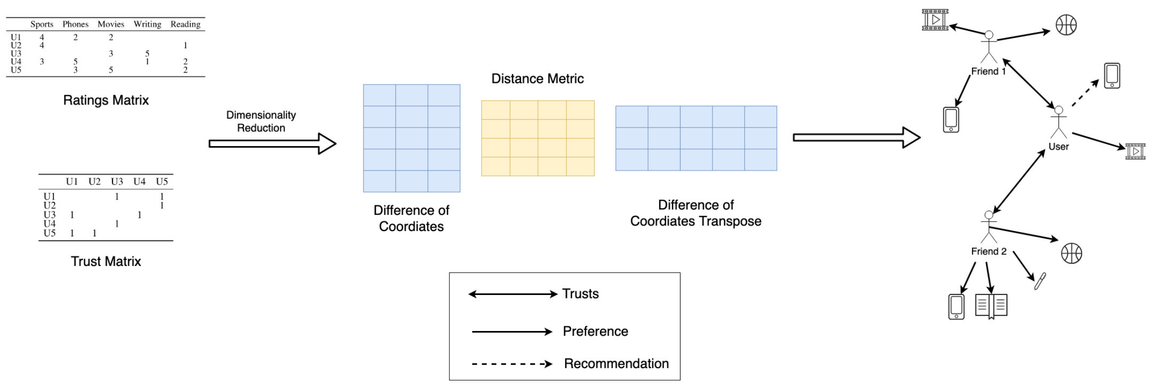
4.2 SocialFD Model

In this sub-section, a social recommender that combines factorization and distance metric learning, also called SocialFD [2] is discussed. Yu et. al. proposed this model to make recommendations more reliable. This model is inspired by the concept "distance reflects likability". With the success of distance metric learning in classification tasks [30], [31], Yu et. al. integrated distance metric with matrix factorization in this model [2]. The main idea of distance metric learning is to "learn a desired distance metric that can make data points with the same class label closer and discriminate data points in different sets with larger distance" [2]. SocialFD model, on the contrary, tries to minimize the distance between each user and his or her friends and items that are rated positively. Also, this algorithm maximizes the distance between users and items rated negatively.

The trust propagation in SocialFD model plays an important role and is similar to that of the collaborative filtering model [1]. Given the information of users' likes and friends where user is denoted by *u*, item by *i*, and friends by *k*, SocialFD also pulls user *k* and item *i* relatively closer in addition to pulling user *u* and item *i* closer. The sparsity problem of user *k* can be overcome by recommending user *u*'s preferred items. Likewise, SocialFD model keeps user *u* away from disliked item *j*, pushing item *j* far from user *k*. Additionally, the SocialFD algorithm decreases the distance between indirect connections or users with similar interests.

One major drawback with the matrix factorization in general is that it is hard to combine a well-trained vector. On contrast, the SocialFD model does flexible inclusion of the ready-made representation of additional knowledge. All the assumptions till now were ratings and social network connections. However, in real-time, user profiles contain huge amounts of texts. These texts can also be accumulated to enhance the quality of recommendations [32] [33].

The graphical illustration of SocialFD model can be seen in Fig. 4 [2]. Users and items seen in Fig. 4 are represented on low-dimensional space. Closer the two symbols are, higher the probability that user prefers that item or trusts another user. Mahalanobis distance is used in this model and is calculated product of latent features difference and the distance metric.



	Sports	Phones	Movies	Writing	Reading
U1	4	2	2		
U2	4				1
U3			3	5	
U4	3	5		1	2
U5		3	5		2

	U1	U2	U3	U4	U5
U1			1		1
U2					1
U3	1			1	
U4			1		
U5	1	1			

Fig. 4. Graphical Representation of SocialFD model [2].

At training stage of the model, constraints are imposed such that users and preferred items or friends are closer and distant from disliked items or users. The ratings and social connections help model to determine the positions of users and items. i.e., if user has expressed only few ratings, his or her social connections or relations can help recommending items to the user. These obtained latent features are interpreted as coordinates and the distance is used to generate meaningful recommendations. The predicted rating in SocialFD model is defined as:

$$\hat{r}_{ui} = \mu + b_u + b_i - \|x_u - y_i\|_A^2 \tag{3}$$

where x_u denotes latent vector of user u, y_i is latent vector of item i, A belongs to distant metric matrix of $k \times k$ dimensions, b_u and b_u are user regularization and item regularization terms respectively. The latent factors are learnt by optimizing the formula:

$$\begin{aligned}\mathscr{L} = &\frac{1}{2}\sum_{u=1}^{m}\sum_{i=1}^{n} N_{u,i}^{R}(r_{ui} - \hat{r}_{ui})^2 \\ &+ \frac{\lambda}{2}\left(\sum_{u=1}^{m} b_u^2 + \sum_{u=1}^{m} b_i^2\right) + \frac{\eta}{2}\left\{\alpha \sum_{(u,i)\in P} \|x_u - y_i\|_A^2 \right. \\ &+ (1-\alpha)\sum_{u\in\mathbf{U}}\sum_{v\in N_u} \|x_u - x_v\|_A^2 \\ &\left. + \alpha \sum_{(u,i)\in N} [1 - \|x_u - y_i\|_A^2]_+ \right\} \qquad (4)\end{aligned}$$

where P is a set of pairs containing user u and his positively rated items, N is the set of pairs containing user u and his negatively rated items, the last three terms are constraints used to adjust the user-user and user-item distance to an appropriate range, λ is to control bias magnitudes, η controls influence of constraints and α manages trade-off between user and item distances.

4.3 GraphRec Model

The GraphRec model is adapted and modified from [3]. This model consists of three components: user modeling, item modeling, and rating prediction [3]. In the user modeling phase, the latent factors users are learnt by the model. There are two aggregations in this component, item aggregation and social aggregation. The item aggregation helps in learning item-space user latent factor from user-item ratings data. This is learned by considering the items that user u has interacted with and the opinions i.e. ratings that u has on these items. The item aggregation in user modeling can be mathematically represented as:

$$h_i^I = \sigma(W.Aggre_{items}(x_{ia}, \forall a \varepsilon C(i)) + b) \qquad (5)$$

where i indicates an item belonging to item set I, C(i) indicates set of item user u has interacted with, x_{ia} is a vector representing opinion-aware interaction between user u and item v_a and $Aggre_{items}$ is an aggregation function. Fig. 5 shows the graphical representation of user modeling phase in GraphRec algorithm.

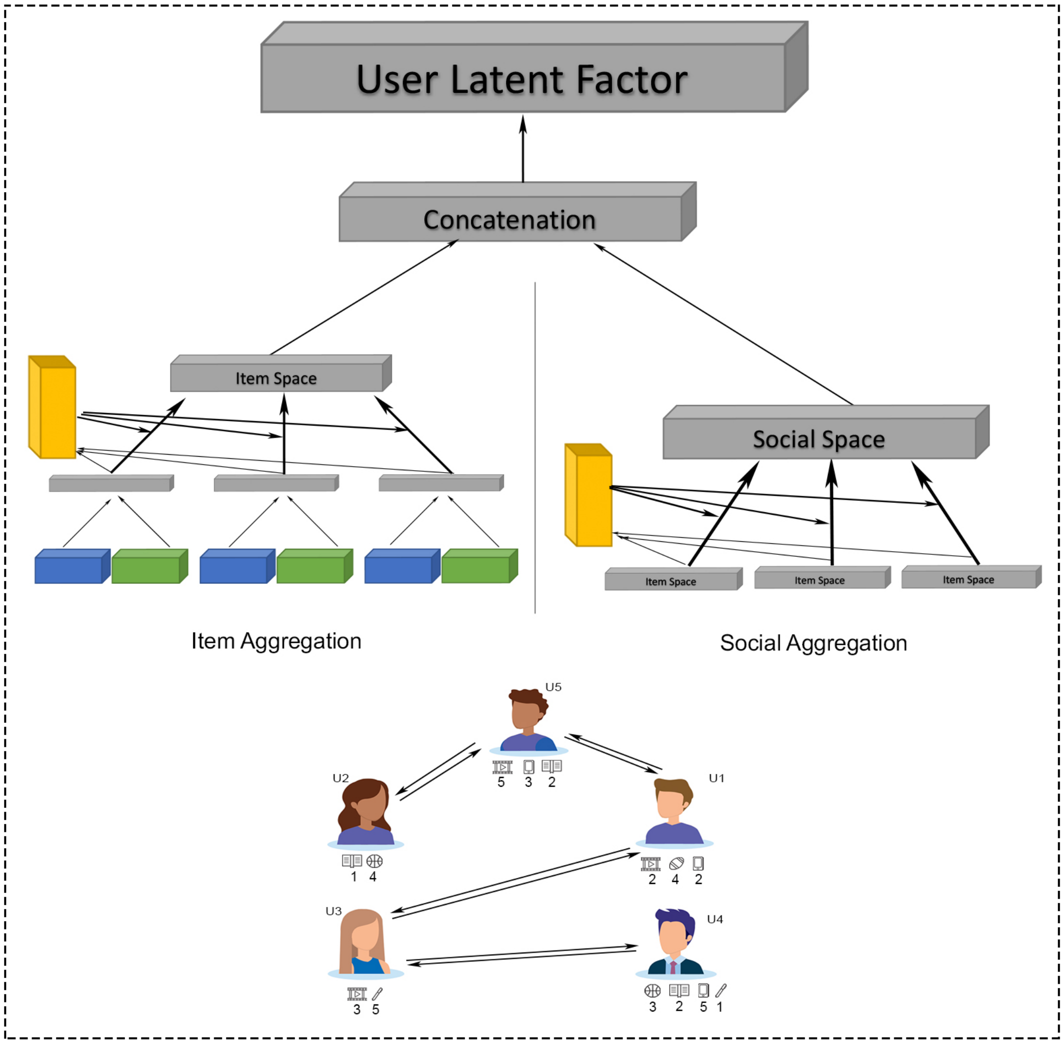


Fig. 5. User modeling phase of GraphRec model [3].

In social aggregation, social-space user latent factor is learned from the social data. According to social correlation theories [34] [35], users opinions towards an item or preferences are either influenced or similar to their direct friends in social networks. In social aggregation of GraphRec, the authors proposed social-space user latent factors, which is to aggregate the item-space user latent factors of neighboring users from the social graph, to incorporate the social correlation theories. Equation 6 denotes the mathematical representation of social aggregation.

$$h_i^S = \sigma(W.Aggre_{neighbors}(h_o^I, \forall o \varepsilon N(i)) + b) \tag{6}$$

Combining the item-space latent factor and social-space latent factor, the total user latent factor is learnt. This can be mathematically represented as equation 7

$$c = [h_i^I \oplus h_i^S] \tag{7}$$

The next component is item modeling. In this component, the item latent factor can be learnt by user aggregation. User aggregation associates item *i* with users that interacted with *i* and their opinions. These opinions or ratings from different users help in capturing the features of same item in different ways provided by users. This helps in modeling item latent factors. Fig. 6 illustrates the item modeling phase graphically and equation 8 gives mathematical representation.

$$z_j = \sigma(W.Aggre_{users}(f_{jt}^I, \forall t \varepsilon B(j)) + b) \tag{8}$$

where B(j) indicates set of users interacted with item v_j, f_{jt} represents opinion-aware interaction of users with item v_j

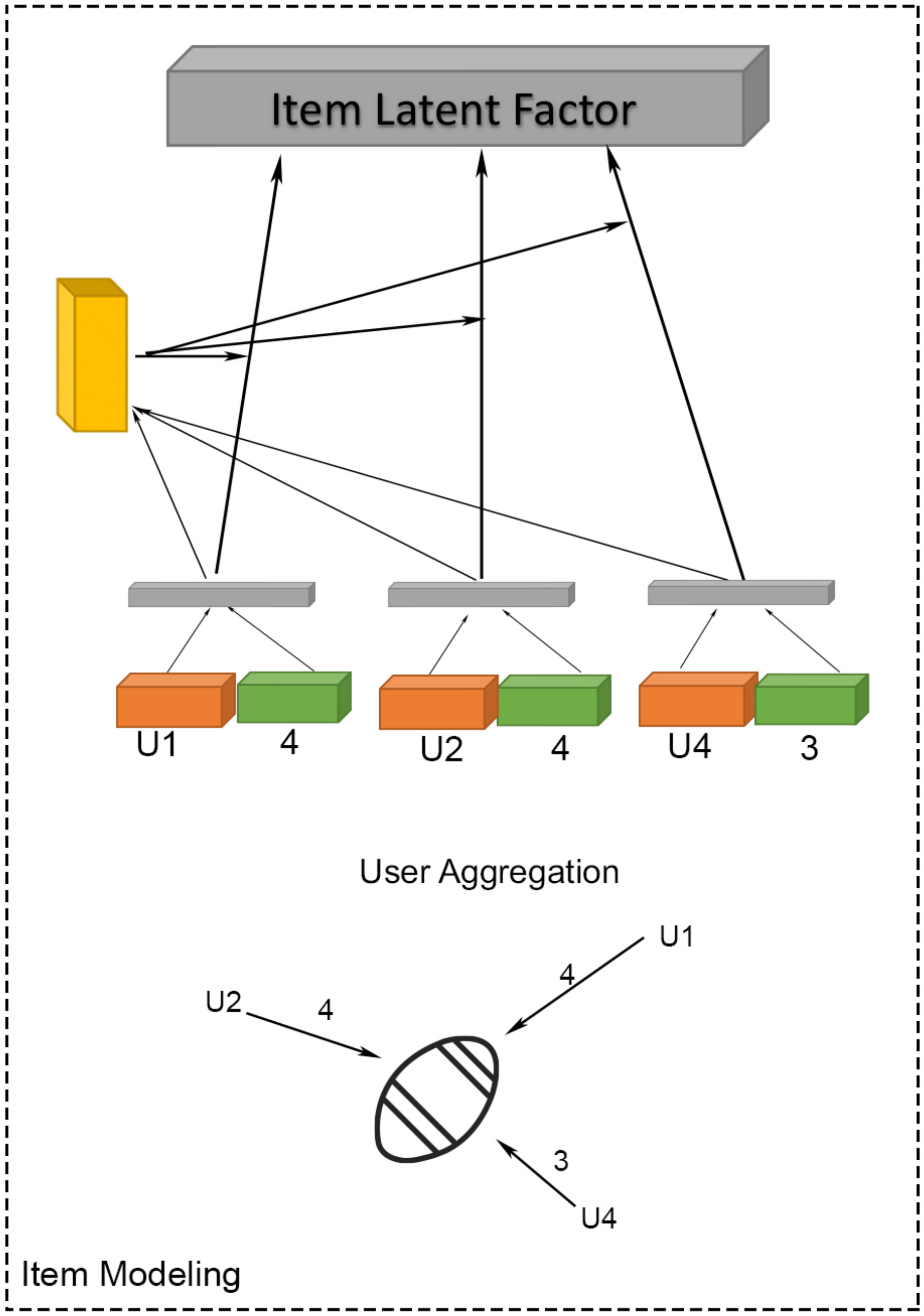


Fig. 6. Item modeling phase of GraphRec model [3].

Finally in the third component, the model parameters are learned and the ratings are predicted using the GraphRec model. The graphical representation of this phase can be seen in Fig. 7. To learn the model parameters, the authors utilized the most commonly used objective function which is formulated as:

$$Loss = \frac{1}{2|O|} \sum_{i,j\varepsilon O} (r'_{ij} - r_{ij})^2 \tag{9}$$

where O is the number of total observed ratings, r_{ij} is rating given by user i to item j.

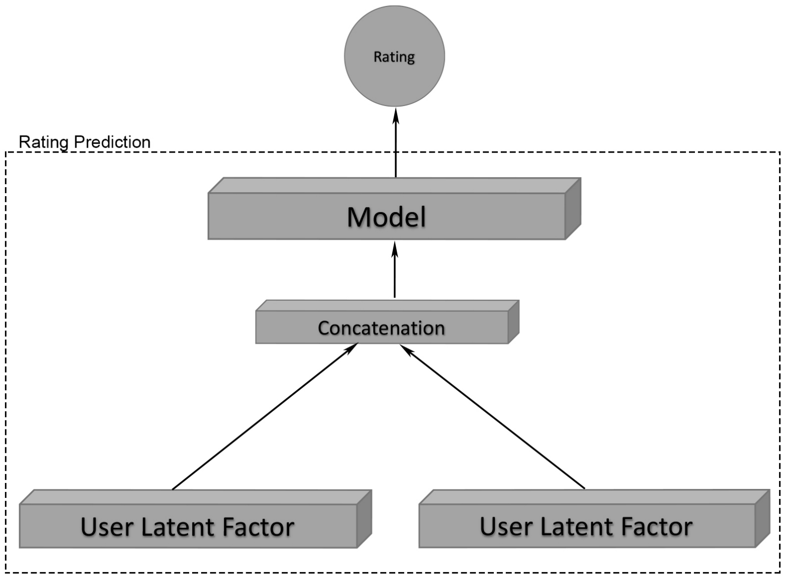


Fig. 7. Rating prediction phase of GraphRec model [3].

For optimization of objective function, the authors used RMSprop defined in [36] rather than the vanilla stochastic gradient descent (SGD). This RMSprop, each time, selects training instance randomly and updates each model parameter towards the direction of its negative gradient [3]. The three embedding item, user and opinions are initialized randomly and learned during the training stage.

The latent features users h_i and items z_j calculated in item modeling and user modeling phases respectively are now concatenated and fed into a Multi-Layer Perceptron

(MLP) for rating prediction. The layers in MLP are as follows:

$$g_1 = [h_i \oplus z_j] \tag{10}$$

where $\oplus$ indicates concatenation

$$g_2 = \sigma(W_2 \cdot g_1 + b_2) \tag{11}$$

$$g_l = \sigma(W_l \cdot g_{l-1} + b_l) \tag{12}$$

rating prediction is done using:

$$r'_{ij} = w^T \cdot g_{l-1} \tag{13}$$

where l is the index of a hidden layer, and r'_{ij} is the predicted rating for item v_j by user u_i.

To avoid overfitting, a persistent problem in optimization of deep neural networks, dropout [37] - a regularization technique for deep neural networks, is utilized. While testing, the dropout regularization is disabled which allows the usage of whole network.

5 EXPERIMENTS

This section gives an overview of the experiments and datasets used in these experiments. Also included are the evaluation metrics used to evaluate these experiments done with SocialMF, SocialFD and GraphRec models.

5.1 Approach

In this paper, our goal is to study how the three representative social recommenders, i.e., SocialMF, SocialFD, and GraphRec, leverage social network information. Towards this goal, we make the hypothesis that if a recommender system truly captures the social network information, making perturbations to the social network should have a significant impact on the performance of these systems. We will study these algorithms on four datasets and five ways to perturb the social network information, as described in the rest of the section.

5.2 Datasets

The major bottleneck in research of social network-based recommender systems is the lack of publicly available social rating network datasets. These models were experimented on with four datasets. Epinions.com is one of the popular publicly available social rating network datasets. For experimentation with these models, we used a version of the Epinions dataset published by the authors of [33]. On average, each user has 8 expressed ratings and has 7 direct neighbors. The next dataset we used is a version of CiaoDVD by the authors of [38]. This dataset is a smaller one compared to the Epinions dataset. Another relevant dataset we used is FilmTrust [39]. FilmTrust is the smallest dataset used in experimentation crawled from the FilmTrust website in 2011.

We have created an additional dataset to experiment with these models. This dataset is an extension of the TwitterEgo dataset by the authors of [40]. The basic dataset consists of social circles from Twitter data which was crawled from public sources. Table 3 shows the compositions of the datasets used for experimentation in this research.

Table 3
Data Statistics

Statistics	CiaoDVD	Epinions	FilmTrust	TwitterEgo
# Users	7,375	40,163	1,508	10,419
# Items	105,114	139,738	2,071	177,558
# Ratings	284,086	664,823	35,497	367,868
# Trust Statements	111,781	487,183	1,853	566,822
Rating Scale	1.0 – 5.0	1.0 – 5.0	1.0 – 5.0	0.0 - 1.0
Average Clustering coefficient	0.0917	0.1449	0.1354	0.3913
Average Closeness centrality	0.0684	0.1478	0.0417	0.1622
Social Trust Density (%)	0.1850%	0.0201%	0.2428%	0.0762%

The density of a social network graph is a measure of the existing number of ties between users compared to number of possible ties between users. This is simply calculated using equation 14 for undirected graphs and equation 15 for directed graphs.

$$UndirectedGraphDensity = \frac{m}{\frac{n(n-1)}{2}} \tag{14}$$

$$DirectedGraphDensity = \frac{m}{n(n-1)} \tag{15}$$

where m denotes cardinality of the social network and n denotes the number of nodes in the social network graph.

The density of a social network helps us understand the comparison of how connected the network is to how connected the social network might be. Also, density helps in differentiating two networks with the same number of nodes and the same type of relationships.

5.3 Social Network Modifications

Taking the original datasets, we modified the social trust data in each of the four datasets for our experiments. This sub-section discusses the modifications we made to these datasets and an example for each.

5.3.1 There is No Trust between any Users - NTU

For this experiment, we modified the social data by removing all the trust statements and providing number of trust statements as 0. i.e. the social network part of data is not considered. The sample social trust matrix would look as in Table 4. Table 5 indicates the change in density of social trust data due to the modifications made.

Table 4
Sample Social Trust Matrix for Users have no Trust

	U1	U2	U3	U4	U5
U1					
U2					
U3					
U4					
U5					

Table 5
Change in Density for NTU

Statistics	CiaoDVD	Epinions	FilmTrust	TwitterEgo
# Users	7,375	40,163	1,508	10,419
# Trust Statements	0	0	0	0
Social Trust Density (%)	0.0%	0.0%	0.0%	0.0%

5.3.2 Users Trust only Themselves - UTU

In this experiment, the social data is modified in such a way that user *u* only trusts or friends with *u* and not anyone else. From the example in Fig. 1, the modified social trust matrix would look as in Table 6. Table 7 indicates the change in density of social trust data due to the modifications made.

Table 6
Sample Social Trust Matrix for Users trust only themselves

	U1	U2	U3	U4	U5
U1	1				
U2		1			
U3			1		
U4				1	
U5					1

Table 7
Change in Density for UTU

Statistics	CiaoDVD	Epinions	FilmTrust	TwitterEgo
# Users	7,375	40,163	1,508	10,419
# Trust Statements	7,375	40,163	1,508	10,419
Social Trust Density (%)	0.0232%	0.0020%	0.1366%	0.0037%

5.3.3 Users Trusts Everyone else Except Themselves - UTEU

In this experiment, the social data is modified in such a way that user u_1 only trusts or friends with all other users in the network U. From the example network in Fig. 1, the modified social trust matrix would look as in Table 8. Table 9 indicates the change in density of social trust data due to the modifications made.

Table 8
Sample Social Trust Matrix for Users trusts everyone else except themselves

	U1	U2	U3	U4	U5
U1		1	1	1	1
U2	1		1	1	1
U3	1	1		1	1
U4	1	1	1		1
U5	1	1	1	1	

Table 9
Change in Density for UTEU

Statistics	CiaoDVD	Epinions	FilmTrust	TwitterEgo
# Users	7,375	40,163	1,508	10,419
# Trust Statements	54,390,625	1,613,026,406	2,272,556	108,545,142
Social Trust Density (%)	100.00%	100.00%	100.00%	100.00%

5.3.4 Users Trust Everyone Including Themselves - UTE

In this experiment, the social data is modified in such a way that user u_1 trusts or friends with every other users in the network U including themselves. From the example social network Fig. 1, the modified social trust matrix would look as in Table 10. Table 11 indicates the change in density of social trust data due to the modifications made.

Table 10
Sample Social Trust Matrix for Users trusts everyone including themselves

	U1	U2	U3	U4	U5
U1	1	1	1	1	1
U2	1	1	1	1	1
U3	1	1	1	1	1
U4	1	1	1	1	1
U5	1	1	1	1	1

Table 11
Change in Density for UTE

Statistics	CiaoDVD	Epinions	FilmTrust	TwitterEgo
# Users	7,375	40,163	1,508	10,419
# Trust Statements	54,390,625	1,613,066,569	2,274,064	108,555,561
Social Trust Density (%)	100.02%	100.06%	100.13%	100.01%

5.4 Evaluation Metrics

In these experiments, MAE and RMSE are chosen to evaluate the quality of recommendations produced by these models.

RMSE is calculated using:

$$RMSE = \sqrt{\frac{1}{n}\Sigma_{u,i}(r_{ui} - r'_{ui})^2} \tag{16}$$

and MAE is defined by:

$$MAE = \frac{1}{n}\Sigma_{u,i}|r_{ui} - r'_{ui}| \tag{17}$$

where n denotes the number of ratings in test set, r_{ui} is the actual rating and r'_{ui} is the predicted rating. Lower MAE and lower RMSE indicate that the missing ratings are predicted more accurately.

5.5 Hyperparameter Tuning

We tuned the hyperparameters such as learning rate, regularization factors (for users, items, and social circles), number of factors, batch size, and number of epochs. All three algorithms are highly dependent on learning rate. These algorithms performed better for learning rates in the range 0.01 to 0.1. The performance of SocialMF and SocialFD algorithms is dependent on social regularization parameter. The ideal values for social regularization parameter are in range 5 to 20. At lower values of social regularization parameter, the performance of SocialMF and SocialFD algorithms is similar to that of the baseline matrix factorization algorithms.

6 RESULTS

In this section, the results of the SocialMF, SocialFD, and GraphRec models are reported for each dataset and compared using the RMSE and MAE evaluation metrics. The SocialMF model results are shown in section 6.1, SocialFD model results in section 6.2, and GraphRec results in section 6.3.

6.1 SocialMF Model Results

Before understanding the results, please note that the code used for implementation of SocialMF algorithm is a modified version provided by the authors of SocialFD [2]. This version has an implementation difference and is mostly in terms of the training phase where the authors of SocialFD have used SGD to obtain the local minimum. The underlying graphical model is still the same. Results of the SocialMF model can be seen in Table 12

Table 12
Results of SocialMF Model on 4 Datasets with Modified Social Trust Information

Experiment	Metrics	CiaoDVD	Epinions	FilmTrust	TwitterEgo
OTD	RMSE	1.0269	1.1044	0.8429	0.1015
	MAE	0.7718	0.8683	0.6389	0.0234
NTU	RMSE	1.0248	1.1455	0.8419	0.0998
	MAE	0.7685	0.8425	0.6344	0.0160
UTU	RMSE	1.0259	1.1037	0.8399	0.1129
	MAE	0.7817	0.8702	0.6389	0.0215
UTEU	RMSE	1.0269	1.1549	0.8521	0.0973
	MAE	0.7785	0.8524	0.6449	0.0159
UTE	RMSE			0.8536	
	MAE			0.6486	

NOTE: Because of the usage of random seed and gradient descent in these experiments, up to two percent difference in RMSE and MAE is negligible.

Due to lack of sufficient memory while unpacking the social network information into a social trust matrix, the "UTE" experiment could not be performed on the CiaoDVD, Epinions, and TwitterEgo datasets. For this reason, the RMSE and MAE are left blank for these datasets on the experiment “UTE.”

The difference of RMSE between experiments “OTD” and “NTU” for Epinions increases to 4%. This increase is slightly higher than our threshold, but the MAE is lower. As graph density changes for each experiment, there might be some effect of this change on the SocialMF model.

Although both SVD-based RS [19] and SocialMF utilize matrix factorization in recommending content or products to users, there are some differences in implementation. In SocialMF, there is an additional step to update the user u’s latent factors based on the user u’s neighbors. This implies that only user matrix is updated in SocialMF and the social trust matrix remains the same. In case of “NTU,” the user matrix does not get updated and the original user matrix is retained. But compared to SVD-based RS, the errors decrease significantly. This might be due to the use of user and item regularization terms. More exploration is needed in understanding how transitivity of trust is affected by changing the social trust information.

Looking at the way the inference is influenced by the trust matrix, we cannot draw any formal conclusions about the impact of the trust matrix on the recommendation. This could be because of multiple reasons. First, based on the fact that social network metrics for these graphs are varied, it may not have anything to do with the social network structure itself. It is, however, possible that the dataset itself is not one in which social influence plays a role. We need further experimentation to verify this formally.

Another reason could be that the modified trust matrices somehow “balanced” out the user latent vectors or the loss function. Details for each modified social graph are below.

In SocialMF, the user latent feature vector is the weighted average of the latent feature vectors of adjacent users in the social graph. The implication of the changed social networks here is that NTU maintains the original latent feature vectors and recommendations are solely based on users that are similar in terms of their ratings. The behavior of NTU is therefore expected to be similar to SVD. However, the presence of the social factor (which is setup as a hyperparameter) might have scaled the recommendation scores.

Extending the argument to the UTU social graph, the user latent vectors are weighted by their own ratings further. We expected that it might have reduced the influence of similar users in the traditional sense to have lesser influence on a user's recommendation. For the UTEU social graph, the user latent vectors are influenced by the average of the latent vectors of other users. For the UTE social graph, the user latent vectors are influenced by the average of all the latent vectors. For both these social graphs, we expected that the average latent vectors might pollute the user similarity with respect to ratings. All of these social graphs were expected to harm the performance of SocialMF. However, the experimental results did not show a significant change in performance.

A key reason for this could be the social trust parameter that tunes the influence of the social network on the recommendation. The authors of SocialMF [1] use a Gaussian prior to determine this factor, which plays a crucial role in their loss function. For our trivial social networks, the priors do not hold. The results, however, do highlight the need to explore the role of the social network further.

In the future, we propose to run further experimentation with random trust matrices to draw more formal conclusions about the role of the social network in the SocialMF algorithm.

6.2 SocialFD Model Results

SocialFD algorithm places users close to their preferred items and friends, and far from their disliked items. The performance of SocialFD model can be seen in Table 13.

From the Tables 12 and 13, it can be inferred that the SocialFD algorithm in general performs better than the SocialMF algorithm consistently for these diverse datasets. This

Table 13
Results of SocialFD Model on 4 Datasets with Modified Social Trust Information

Experiment	Metrics	CiaoDVD	Epinions	FilmTrust	TwitterEgo
OTD	RMSE	0.9645	1.0458	0.7806	0.0221
	MAE	0.7261	0.7932	0.5948	0.0159
NTU	RMSE	0.9641	1.0456	0.7803	0.0161
	MAE	0.7260	0.7906	0.5947	0.0069
UTU	RMSE	0.9594	1.0485	0.7709	0.0198
	MAE	0.7368	0.7899	0.6008	0.0079
UTEU	RMSE	0.9590	1.0548	0.7702	0.0159
	MAE	0.7160	0.7897	0.5893	0.0068
UTE	RMSE			0.7904	
	MAE			0.6039	

indicates that distance metrics may have an important role in social recommendations. However, when we compare these models between "OTD" and "NTU", there is no significant change in the RMSE and MAE. In some cases, there seems to be a marginal improvement without trust information. Looking back at our hypothesis in Section 5.1, the results indicate that there is a need for deeper exploration on these lines.

6.3 GraphRec Model Results

GraphRec is a deep learning model that uses attention and graph neural networks. The results obtained by using GraphRec model can be seen in the Table 14.

GraphRec showed improved performance while considering social trust information when compared to other models like SocialMF [1], SoRec [15], etc. Although GraphRec is a more complex deep learning algorithm, we did not observe much difference in RMSE and MAE as compared to SocialFD. In some cases, these errors increased significantly. For example, for our TwitterEgo dataset, the MAE increased significantly and for the FilmTrust dataset both metrics increased. We also observed that for the same model, performance improves in some scenarios and deteriorates in the others. Regardless, the difference in performance is very little. This highlights the fact that deep learning

Table 14
Results of GraphRec Model on 4 Datasets with Modified Social Trust Information

Experiment	Metrics	CiaoDVD	Epinions	FilmTrust	TwitterEgo
OTD	RMSE	1.0022	1.0818	0.9057	0.0209
	MAE	0.7611	0.8299	0.6970	0.0149
NTU	RMSE	0.9989	1.0786	0.9051	0.0194
	MAE	0.7645	0.8255	0.6921	0.0140
UTU	RMSE	1.0342	1.1008	0.8975	0.0198
	MAE	0.7903	0.8382	0.7108	0.0151
UTEU	RMSE	0.9689	1.0983	0.9005	0.0182
	MAE	0.7945	0.8356	0.6891	0.0137
UTE	RMSE			0.9063	
	MAE			0.6985	

techniques cannot guarantee a better performance even with huge datasets such as Epinions and TwitterEgo. It is particularly interesting that in most cases, the trivial social trust datasets perform better than the default “OTD” datasets.

6.4 Performance Comparison and Summary

The RMSE and MAE of TwitterEgo dataset are low compared to other datasets because its rating scale is binary, i.e., 0 or 1.

Fig. 8 illustrates the performance of these three models on the CiaoDVD dataset. It can be observed that the performance of each algorithm is similar in all the experiments. Similar to CiaoDVD, the results for the Epinions (Fig. 9) and FilmTrust (Fig. 10), datasets show that performance of these algorithms is similar for all perturbations on the social trust data. Whereas, for the TwitterEgo dataset (Fig. 11), it can be seen that the MAE for “OTD” experiment is higher for SocialMF and SocialFD models.

All of the above experiments indicate the following key observations. Comparing social recommendation algorithm with trust information indicates that SocialFD is a superior algorithm for all the datasets and social trust data compared to SocialMF. However, from our experiments, social trust information does not significantly improve the performance of these representative models on diverse datasets. Also, we understand that superior performance of SocialFD algorithm has something to do with the utilization of distance metric. Further study is needed in the direction of incorporating distance metric learning into social recommender systems. The computational overhead and complexity of deep learning models may not make a significant difference to the performance. It also highlights the importance of studying the interpretability of deep learning social recommender systems in general.

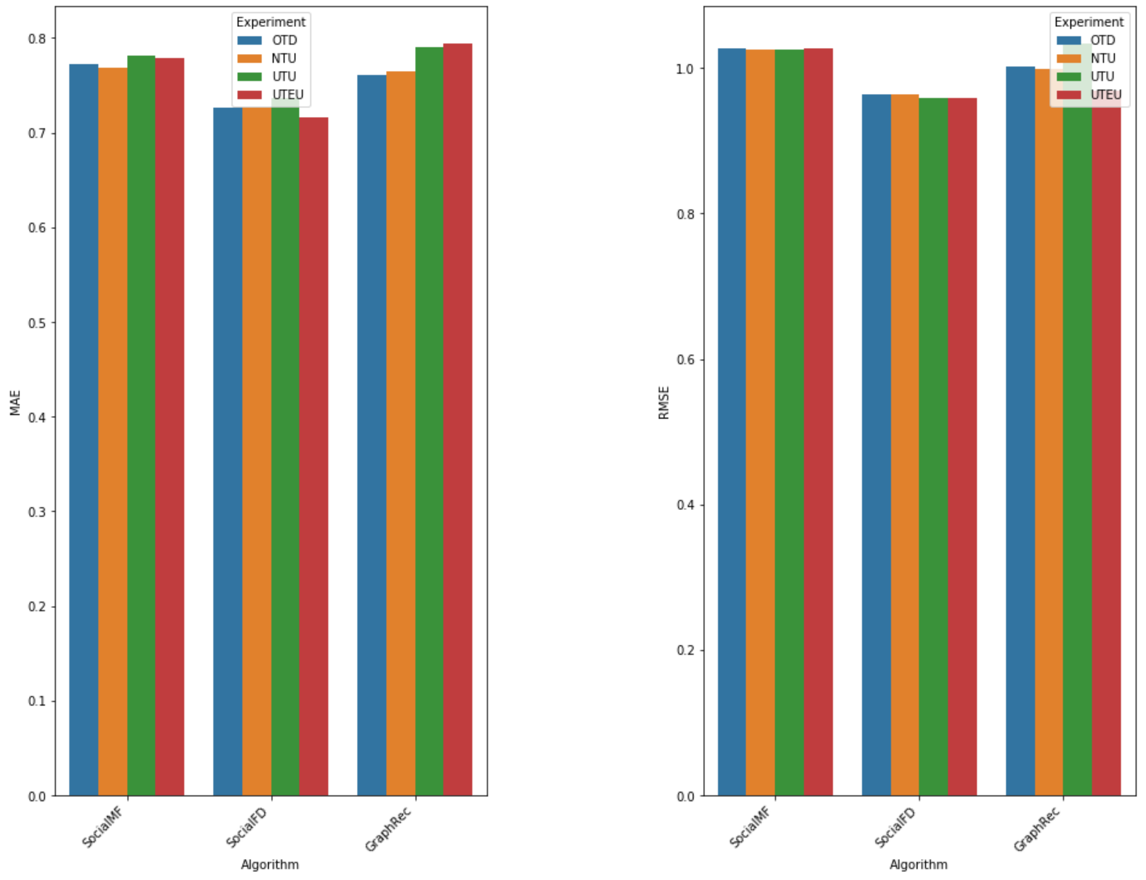
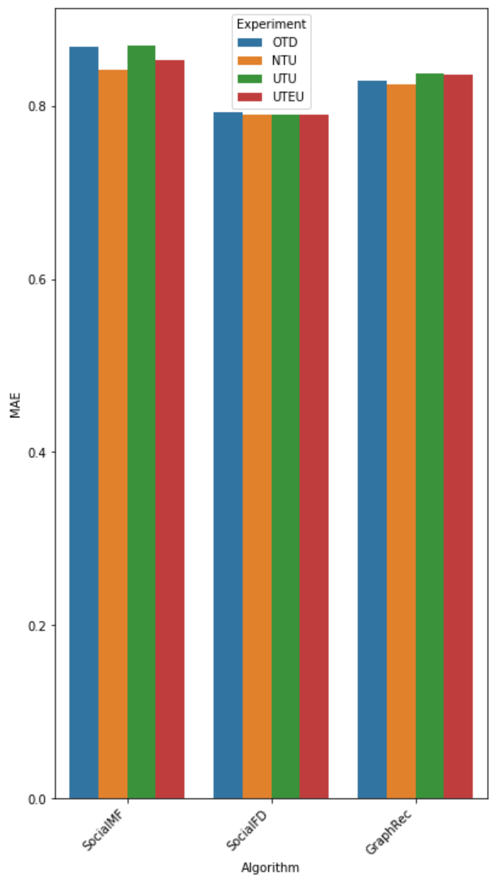


Fig. 8. Graphs comparing results of experiments on CiaoDVD dataset.



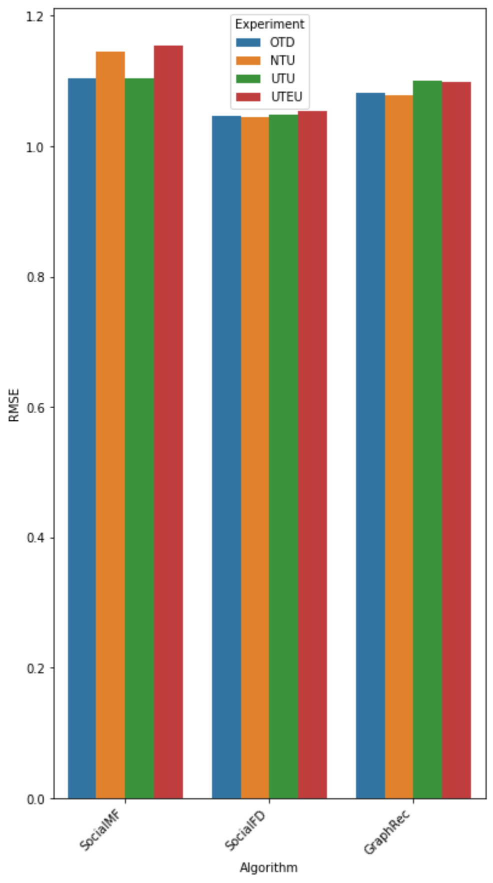


Fig. 9. Graphs comparing results of experiments on Epinions dataset.

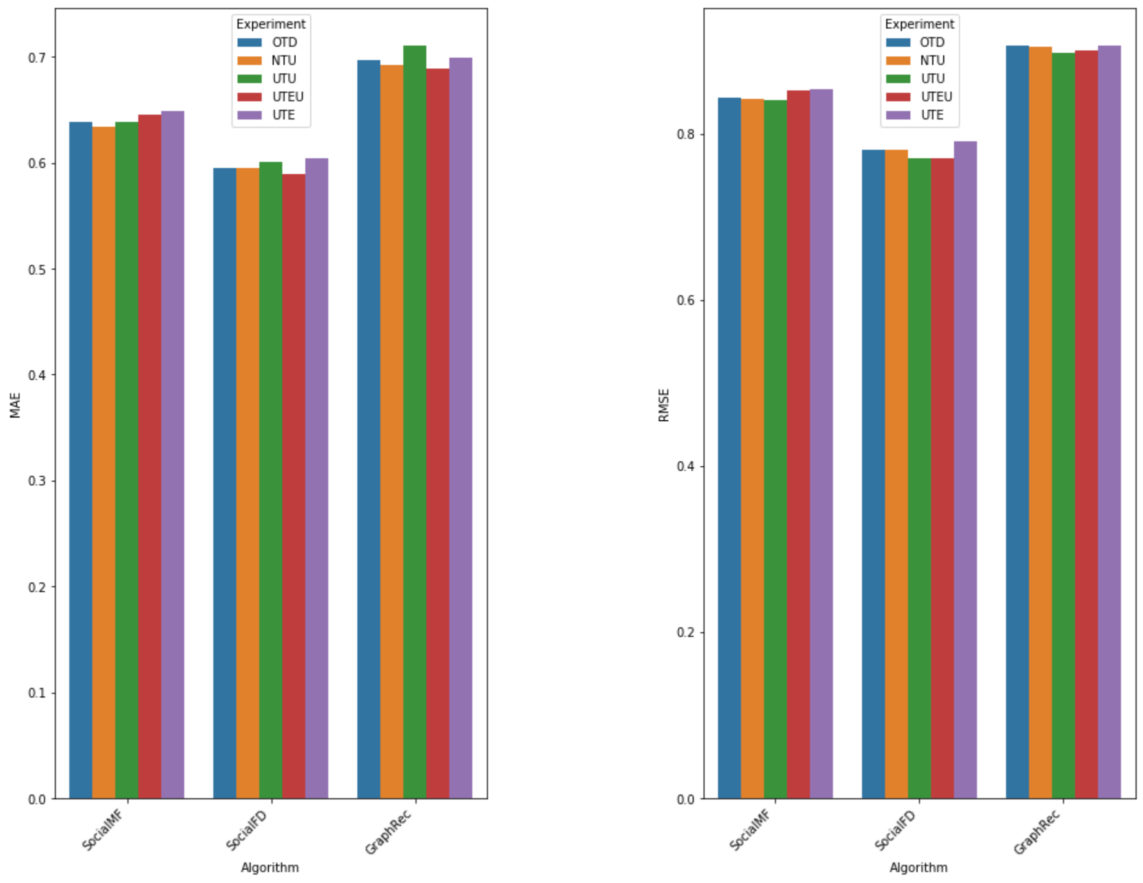


Fig. 10. Graphs comparing results of experiments on FilmTrust dataset.

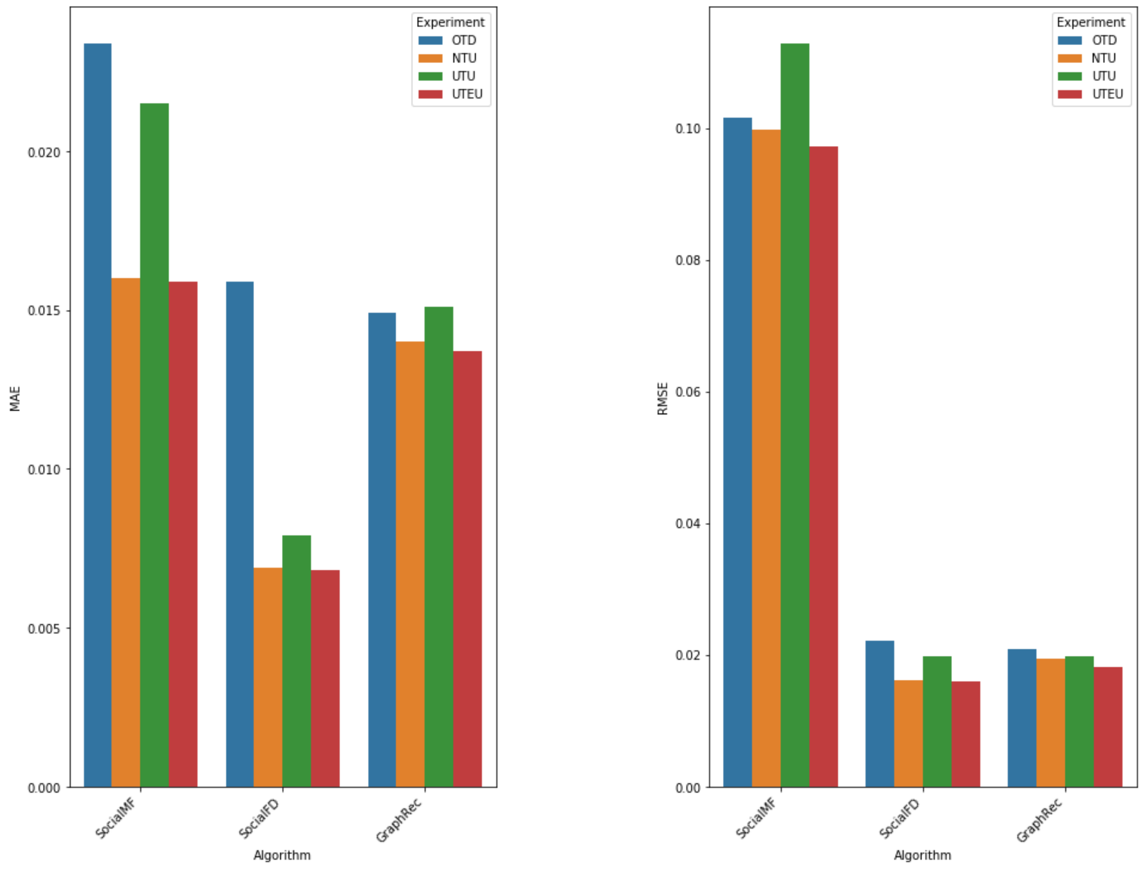


Fig. 11. Graphs comparing results of experiments on TwitterEgo dataset.

7 CONCLUSIONS

Recommender systems are powerful tools that filter and recommend content/information relevant to a given user. With the advent of social networks, it has become very important to utilize the data on social ties between in a social network to recommend a product to the users who expressed a few ratings. In this paper, we explored three such social recommender models, SocialMF, SocialFD, and GraphRec. SocialMF is a model that incorporates trust propagation into matrix factorization, SocialFD model uses distance metric learning in addition to matrix factorization, and GraphRec is a model that uses attenttion-based graph neural networks.

Experiments on 4 real life datasets, CiaoDVD, Epinions, FilmTrust, and TwitterEgo, show that these algorithms outperform the conventional collaborative filtering algorithms as well as the previously developed social recommender systems. Of these three, the SocialFD model performs better than the SocialMF model with inclusion of trust. At lower social parameter values, these models' performance is similar to the performance of collaborative filtering algorithms. However, when social trust factor is not given, these models show similar performance compared to the models that contain social trust parameter. From these experiments, it can be seen that there is less conclusive evidence that social recommender systems are influenced by social trust data. The experiments highlight the need to explore further to gain better understanding of the role of social networks in recommender systems.

8 FUTURE WORK

This work suggests some interesting directions for future research. These models can be extended further to handle zero and negative trust relations. In general, negative trust, also called distrust, gives more information about a user than positive opinions. Also, currently, social regularization parameter is given as an input to these models. Future work can help in the development of a model that incorporates automatic tuning of social trust. In real-time, user profiles and item profiles contain huge amounts of text data and other features. These features can also be accumulated into the recommender system to enhance the quality of recommendations. In the future, we would like to explore a dataset where social network is explicitly synthesized to have an impact on recommendation and repeat these experiments on that synthetic dataset.

Literature Cited

[1] M. Jamali and M. Ester, “A matrix factorization technique with trust propagation for recommendation in social networks,” in *Proceedings of the Fourth ACM Conference on Recommender Systems*, RecSys ’10, (New York, NY, USA), p. 135–142, Association for Computing Machinery, 2010.

[2] J. Yu, M. Gao, W. Rong, Y. Song, and Q. Xiong, “A social recommender based on factorization and distance metric learning,” *IEEE Access*, vol. 5, pp. 21557–21566, 2017.

[3] W. Fan, Y. Ma, Q. Li, Y. He, E. Zhao, J. Tang, and D. Yin, “Graph neural networks for social recommendation,” WWW ’19, (New York, NY, USA), p. 417–426, Association for Computing Machinery, 2019.

[4] F. Ricci, L. Rokach, and B. Shapira, *Introduction to Recommender Systems Handbook*, pp. 1–35. Boston, MA: Springer US, 2011.

[5] L. Terveen and W. Hill, “Beyond recommender systems: Helping people help each other,” 2001.

[6] D. Goldberg, D. Nichols, B. M. Oki, and D. Terry, “Using collaborative filtering to weave an information tapestry,” *Commun. ACM*, vol. 35, p. 61–70, Dec. 1992.

[7] A. S. Das, M. Datar, A. Garg, and S. Rajaram, “Google news personalization: Scalable online collaborative filtering,” in *Proceedings of the 16th International Conference on World Wide Web*, WWW ’07, (New York, NY, USA), p. 271–280, Association for Computing Machinery, 2007.

[8] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, “Neural collaborative filtering,” in *Proceedings of the 26th International Conference on World Wide Web*, WWW ’17, (Republic and Canton of Geneva, CHE), p. 173–182, International World Wide Web Conferences Steering Committee, 2017.

[9] D. Liang, R. G. Krishnan, M. D. Hoffman, and T. Jebara, “Variational autoencoders for collaborative filtering,” in *Proceedings of the 2018 World Wide Web Conference*, WWW ’18, (Republic and Canton of Geneva, CHE), p. 689–698, International World Wide Web Conferences Steering Committee, 2018.

[10] M. F. Dacrema, P. Cremonesi, and D. Jannach, “Are we really making much progress? a worrying analysis of recent neural recommendation approaches,” RecSys ’19, (New York, NY, USA), p. 101–109, Association for Computing Machinery, 2019.

[11] X. Qian, H. Feng, G. Zhao, and T. Mei, “Personalized recommendation combining user interest and social circle,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, no. 7, pp. 1763–1777, 2014.

[12] J. Golbeck, *Computing and Applying Trust in Web-based Social Network*. PhD thesis, University of Maryland, College Park, MD, 2005.

[13] M. Jamali and M. Ester, “Trustwalker: A random walk model for combining trust-based and item-based recommendation,” in *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD ’09, (New York, NY, USA), p. 397–406, Association for Computing Machinery, 2009.

[14] H. Ma, I. King, and M. R. Lyu, “Learning to recommend with social trust ensemble,” SIGIR ’09, (New York, NY, USA), p. 203–210, Association for Computing Machinery, 2009.

[15] H. Ma, H. Yang, M. R. Lyu, and I. King, “Sorec: Social recommendation using probabilistic matrix factorization,” in *Proceedings of the 17th ACM Conference on Information and Knowledge Management*, CIKM ’08, (New York, NY, USA), p. 931–940, Association for Computing Machinery, 2008.

[16] P. Massa and P. Avesani, “Trust-aware recommender systems,” RecSys ’07, (New York, NY, USA), p. 17–24, Association for Computing Machinery, 2007.

[17] C. Ziegler, *Towards decentralized recommender systems*. PhD thesis, University of Freiburg, 2005.

[18] Y. Koren, “Factorization meets the neighborhood: A multifaceted collaborative filtering model,” KDD ’08, (New York, NY, USA), p. 426–434, Association for Computing Machinery, 2008.

[19] Y. Koren, R. Bell, and C. Volinsky, “Matrix factorization techniques for recommender systems,” *Computer*, vol. 42, no. 8, pp. 30–37, 2009.

[20] M. Richardson and P. Domingos, “Mining knowledge-sharing sites for viral marketing,” KDD ’02, (New York, NY, USA), p. 61–70, Association for Computing Machinery, 2002.

[21] R. Levien, *Attack-Resistant Trust Metrics*, pp. 121–132. London: Springer London, 2009.

[22] T. N. Kipf and M. Welling, “Semi-supervised classification with graph convolutional networks,” 2017.

[23] M. Defferrard, X. Bresson, and P. Vandergheynst, “Convolutional neural networks on graphs with fast localized spectral filtering,” NIPS’16, (Red Hook, NY, USA), p. 3844–3852, Curran Associates Inc., 2016.

[24] W. L. Hamilton, R. Ying, and J. Leskovec, “Inductive representation learning on large graphs,” NIPS’17, (Red Hook, NY, USA), p. 1025–1035, Curran Associates Inc., 2017.

[25] Y. Ma, S. Wang, C. Aggarwal, D. Yin, and J. Tang, “Multi-dimensional graph convolutional networks,” in *SIAM International Conference on Data Mining, SDM 2019*, SIAM International Conference on Data Mining, SDM 2019, (United States), pp. 657–665, Society for Industrial and Applied Mathematics Publications, 2019. 19th SIAM International Conference on Data Mining, SDM 2019 ; Conference date: 02-05-2019 Through 04-05-2019.

[26] Q. Wu, H. Zhang, X. Gao, P. He, P. Weng, H. Gao, and G. Chen, “Dual graph attention networks for deep latent representation of multifaceted social effects in recommender systems,” WWW ’19, (New York, NY, USA), Association for Computing Machinery, 2019.

[27] G. Bathla, H. Aggarwal, and R. Rani, “Autotrustrec: Recommender system with social trust and deep learning using autoencoder,” *Multimedia Tools and Applications*, vol. 79, 08 2020.

[28] H. Ma, D. Zhou, C. Liu, M. R. Lyu, and I. King, “Recommender systems with social regularization,” in *Proceedings of the fourth ACM international conference on Web search and data mining*, pp. 287–296, 2011.

[29] G. Guo, J. Zhang, and N. Yorke-Smith, “A novel bayesian similarity measure for recommender systems,” IJCAI ’13, p. 2619–2625, AAAI Press, 2013.

[30] J. Goldberger, G. E. Hinton, S. T. Roweis, and R. R. Salakhutdinov, "Neighbourhood components analysis," in *Advances in Neural Information Processing Systems 17* (L. K. Saul, Y. Weiss, and L. Bottou, eds.), pp. 513–520, MIT Press, 2005.

[31] K. Q. Weinberger and L. K. Saul, "Distance metric learning for large margin nearest neighbor classification," *Journal of Machine Learning Research*, vol. 10, no. 9, pp. 207–244, 2009.

[32] C. Musto, "Enhanced vector space models for content-based recommender systems," RecSys '10, (New York, NY, USA), p. 361–364, Association for Computing Machinery, 2010.

[33] G. Semeraro, P. Lops, P. Basile, and M. de Gemmis, "Knowledge infusion into content-based recommender systems," RecSys '09, (New York, NY, USA), p. 301–304, Association for Computing Machinery, 2009.

[34] P. V. MARSDEN and N. E. FRIEDKIN, "Network studies of social influence," *Sociological Methods & Research*, vol. 22, no. 1, pp. 127–151, 1993.

[35] M. McPherson, L. Smith-Lovin, and J. M. Cook, "Birds of a feather: Homophily in social networks," *Review of Sociology*, vol. 27, pp. 415–444, 2001.

[36] T. Tieleman and G. Hinton, "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude," *COURSERA: Neural networks for machine learning*, vol. 4, no. 2, pp. 26–31, 2012.

[37] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 56, pp. 1929–1958, 2014.

[38] G. Zhao, X. Qian, and X. Xie, "User-service rating prediction by exploring social users' rating behaviors," *IEEE Transactions on Multimedia*, vol. 18, no. 3, pp. 496–506, 2016.

[39] J. Tang, H. Gao, and H. Liu, "Mtrust: Discerning multi-faceted trust in a connected world," in *Proceedings of the Fifth ACM International Conference on Web Search and Data Mining*, WSDM '12, (New York, NY, USA), p. 93–102, Association for Computing Machinery, 2012.

[40] J. Leskovec and J. J. Mcauley, "Learning to discover social circles in ego networks," in *Advances in Neural Information Processing Systems 25* (F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, eds.), pp. 539–547, Curran Associates, Inc., 2012.

[41] G. Sabidussi, "The centrality index of a graph," *Psychometrika*, vol. 31, no. 4, pp. 581–603, 1966.

Appendix A

GENERATION OF EXTENDED TWITTEREGO DATASET

This dataset is an extension of TwitterEgo dataset by authors of [40]. The basic dataset consists of social circles information from Twitter data which was crawled from public sources. Using an open source API known as “Tweepy”, we extracted tweets to which each of the users reacted. Using the tweet IDs, the tweets to which users reacted are rated as 1 and the tweets to which the users did not react were rated as 0. Finally, we created a ratings dataset for the users and tweets. We generated the social relations dataset using the social circles from the original dataset. We divided the data into train, validation and test using stratified split technique.

This dataset can be downloaded from this repository. The statistics of TwitterEgo dataset are:

Table 15
TwitterEgo Data Statistics

# Users	10,419
# Items	177,558
# Ratings	367,868
# Trust Statements	566,822
Rating Scale	0.0 - 1.0
Average Ratings per user	35
Average clustering coefficient	0.3913
Average closeness centrality	0.1622

In a graph network, **closeness centrality** is a measure used to determine the centrality of a node. It is calculated as ”the reciprocal of the sum of the length of the shortest paths between the node and all other nodes in the graph” [41]. A node is *closer* to all other nodes if it is more central.

Appendix B

PREPROCESSING FOR GRAPHREC

Although the authors of GraphRec model [3] made their code public, one problem we faced was preprocessing data to match the model requirements. Using Python 3.6 version, we created a script to preprocess the data. This script performs the following actions:

- Ratings data
  - Each user is associated with all the items that he/she has rated/preferred/reacted to
  - For each user, the ratings that the user has given are also associated
  - Each item is associated with all the user that rated/preferred/reacted to the item
  - For each item, the ratings that the item has been given are also associated
- Social Trust Data
  - Each user is aggregated with all the other users that he/she trusts
- Based on the number of ratings, opinion embedding with 5 different embedding vectors are randomly initialized. i.e., for a rating scale of 1 to 5, embedding vectors are randomly initialized based on 5 scores in 1, 2, 3, 4, 5

The entire data, which is in form of Python objects is stored as a byte stream through a process called "pickling". This is done using a Python module called "pickle" which implements binary protocols for serializing and de-serializing a Python object structure.

The "dataset.pickle" file is then loaded to the GraphRec model to calculate, the user-latent factors, item-latent factors, and perform predictions on the test data.

Appendix C

MODIFIED GRAPHREC VERSION

While experimenting with the GraphRec model using original source code provided by the authors, we had to modify it to fit needs of our datasets and preprocessing techniques. Apart from these changes, we modified the architecture of the GraphRec model. The architecture of modified model can be seen in Figure 12

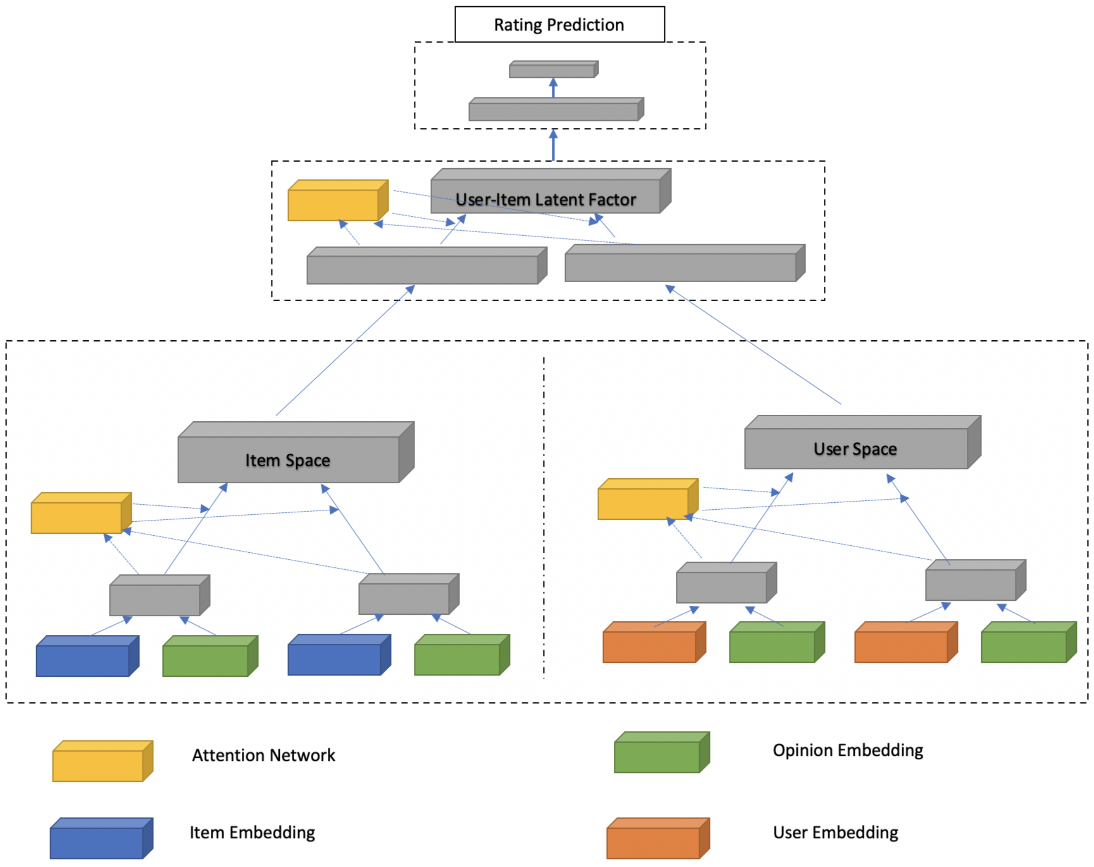


Fig. 12. Architecture of modified GraphRec model

In this version, both the user aggregation and item item aggregation are performed in first step. The item space obtained in user aggregation step and user space obtained in item aggregation step are then concatenated and passed as input to social aggregation step.

In social aggregation step, user-item latent factors are calculated. These latent factors are then used to train the model and make predictions. Table 16 shows the performance of this modified version of GraphRec algorithm.

Table 16
Results of Modified GraphRec Model on 4 Datasets with Modified Social Trust Information

Experiment	Metrics	CiaoDVD	Epinions	FilmTrust
OTD	RMSE	1.0052	1.0759	0.9050
	MAE	0.7714	0.8367	0.6979
NTU	RMSE	0.9890	1.0668	0.9034
	MAE	0.7592	0.8295	0.6891